IFT2015 hiver 2010 — Notes de cours

Miklós Csűrös

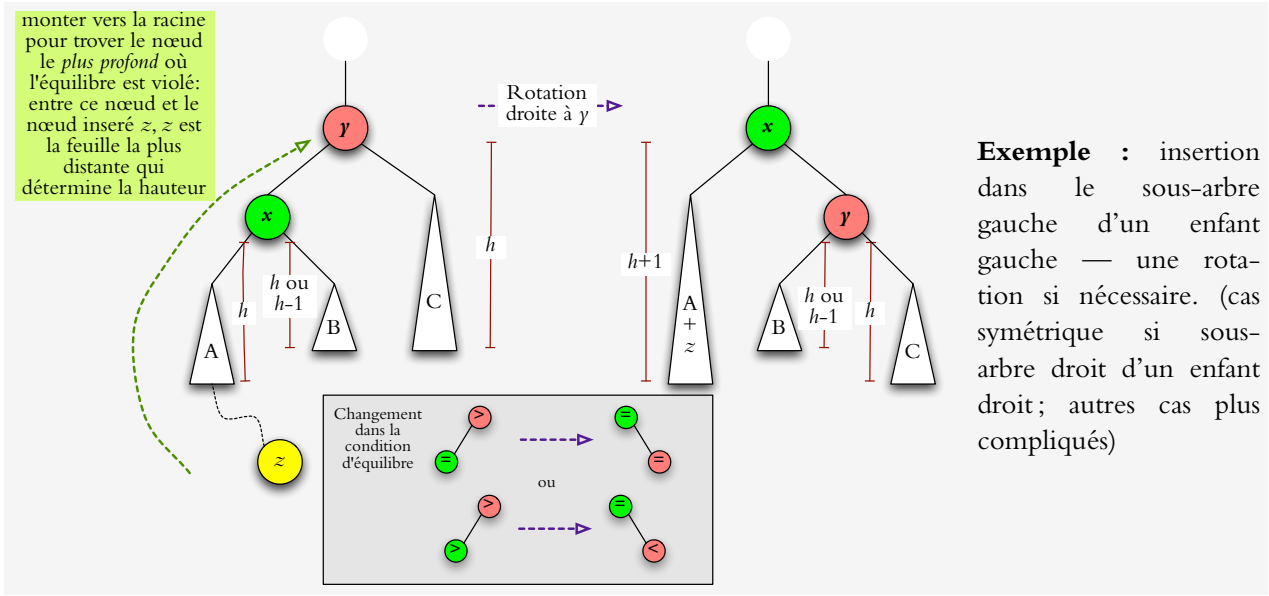
21 mars 2010

15 ABRs équilibrés

15.1 Arbre AVL [Adelson-Velsky et Landis (1962)]

Définition 15.1. *Un arbre binaire est un arbre AVL ssi à chaque nœud, la hauteur du sous-arbre gauche et la hauteur du sous-arbre droit diffèrent par 1 au plus. (On considère la hauteur d'un sous-arbre vide = -1.)*

Pour vérifier l'équilibre AVL, on peut stocker la hauteur de chaque sous-arbre à sa racine. Dans une implantation il suffit d'indiquer le cas applicable correspondant à une des trois relations possibles $<$, $=$, $>$ entre les hauteurs des sous-arbres. Lors d'une insertion (ou suppression), il faut rétablir l'équilibre en performant $O(1)$ (ou $O(\log n)$) rotations.



Exemple : insertion dans le sous-arbre gauche d'un enfant gauche — une rotation si nécessaire. (cas symétrique si sous-arbre droit d'un enfant droit ; autres cas plus compliqués)

Lemme 15.1. *Soit $N(h)$ le nombre minimal de nœuds dans un arbre AVL de hauteur $h \geq 0$. On a $N(0) = 1$, $N(1) = 2$. Pour tout $h > 1$, $N(h) = N(h-1) + N(h-2) + 1$.*

Exercice 15.1. *Dessinez les arbres AVL à hauteur maximale et minimale pour 7, 12 et 20 nœuds.*

Lemme 15.2. *Il existe $c > 0$ tel que $N(h) \leq c\phi^h - 1$ pour tout $h \geq 0$ où $\phi = \frac{1+\sqrt{5}}{2}$. (Notez que $\phi^2 = \phi + 1$.)*

Démonstration. La constante c sera spécifiée plus tard. Supposons que $N(k) \leq c\phi^k - 1$ pour tout $0 \leq k < h$. Alors,

$$N(h) = N(h-1) + N(h-2) + 1 \leq c\phi^{h-1} + c\phi^{h-2} - 1 = c\phi^h - 1.$$

Si on choisit $c = 2$, la borne est correcte pour $h = 0, 1$ et donc elle est correcte pour tout h. ■

Théorème 15.3. *La hauteur h d'un arbre AVL avec n nœuds [non-null] est bornée comme*

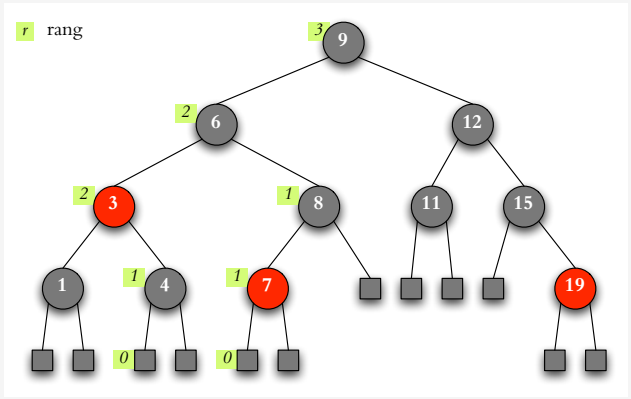
$$\lg(n+1) - 1 \leq h \leq \log_\phi \frac{n+1}{2} = \frac{\lg(n+1) - 1}{\lg \phi} \approx 1.44 \lg n. \tag{15.1}$$

En conséquence, $h = \Theta(\log n)$ même dans le pire des cas.

15.2 Arbre rouge et noir

On associe un **rang** à chaque nœud, qui est une valeur entière et non-négative. Notation : $\mathsf{rang}(x)$.

1. Pour chaque nœud x excepté la racine,
$$\mathsf{rang}(x) \leq \mathsf{rang}(\mathsf{parent}(x)) \leq \mathsf{rang}(x) + 1.$$
2. Pour chaque nœud x avec grand-parent $y = \mathsf{parent}(\mathsf{parent}(x))$,
$$\mathsf{rang}(x) < \mathsf{rang}(y).$$
3. Pour chaque feuille (null) on a $\mathsf{rang}(x) = 0$ et $\mathsf{rang}(\mathsf{parent}(x)) = 1$.



Au lieu de stocker les rangs explicitement, il suffit de colorier les nœuds par rouge ou noir.

- ⋆ si $\mathsf{rang}(\mathsf{parent}(x)) = \mathsf{rang}(x)$, alors x est colorié par **rouge**
- ⋆ si x est la racine ou $\mathsf{rang}(\mathsf{parent}(x)) = \mathsf{rang}(x)+1$, alors x est colorié par **noir**

Théorème 15.4. *Dans un coloriage valide,*
(0) chaque nœud est soit noir soit rouge
*(i) chaque feuille (*null*) est noire*
(ii) le parent d'un nœud rouge est noir
(iii) chaque chemin reliant un nœud à une feuille dans son sous-arbre contient le même nombre de nœuds noirs

En (iii), le nombre de nœuds noirs sur le chemin est égal au rang ⇒ rang est parfois appelé **hauteur noire**.

Lemme 15.5. *Pour chaque nœud x, sa hauteur $h(x) \leq 2 \cdot \mathsf{rang}(x)$.*

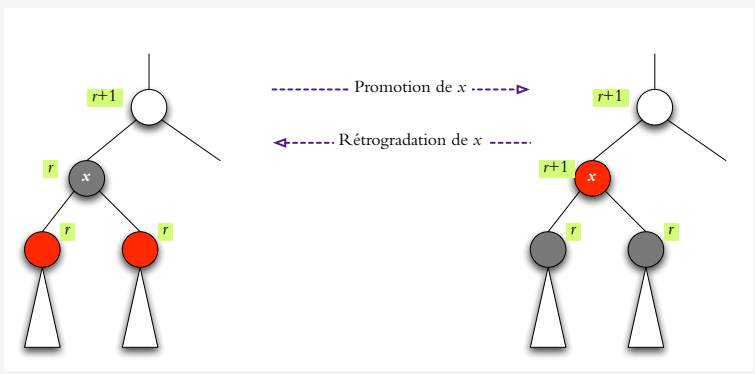
Démonstration. On doit avoir au moins autant de nœuds noirs que de nœuds rouges dans un chemin à une feuille. ■

Lemme 15.6. *Le nombre de déscendants internes de chaque nœud x est $\geq 2^{\mathsf{rang}(x)} - 1$.*

Démonstration. Par induction. Le théorème est vrai pour une feuille x quand $\mathsf{rang}(x) = 0$. Supposons que le théorème est vrai pour tout x avec une hauteur $h(x) < k$. Considérons un nœud x avec $h(x) = k$ et ses deux enfants u, v avec $h(u), h(v) < k$. Par l'hypothèse d'induction, le nombre des descendants de x est $\geq 1 + (2^{\mathsf{rang}(u)} - 1) + (2^{\mathsf{rang}(v)} - 1)$. Or, $\mathsf{rang}(x) - 1 \leq \mathsf{rang}(u), \mathsf{rang}(v)$. ■

Théorème 15.7. *La hauteur d'un arbre RN avec n nœuds non-null est bornée comme*

$$\lg(n+1) - 1 \leq h \leq 2\lg(n+1). \tag{15.2}$$



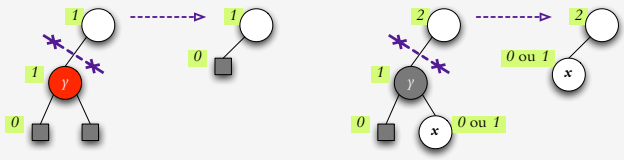
Pour maintenir l'équilibre, on utilise les **rotations** comme avant + **promotion/rétrogradation** (incrémenter ou décrementer le rang par 1).
→ promotion/rétrogradation change la couleur d'un nœud et ses enfants.
→ on ne peut promouvoir x que s'il est noir avec deux enfants rouges

15.3 Insertion dans l'arbre RN

On insère x avec $\mathsf{rang}(x) = 1 \Rightarrow$ sa couleur est rouge.

Test : est-ce que le parent de x est rouge ? Si oui, on a un problème ; sinon, rien à faire (cas 0a). Solution : soit $y = \mathsf{parent}(\mathsf{parent}(x))$ le grand-parent — il est noir. Si y a deux enfants rouge, alors promouvoir y et retourner au test avec $x \leftarrow y$ (cas 0b). Si on a fini les promotions et il y a toujours le problème que x est rouge, son parent est rouge aussi, mais l'oncle de x est noir. On fait une ou deux rotations (cas 1 ou 2) selon la relation entre x et $\mathsf{parent}(x)$ et leurs parents (comme les cas zig-zag, zag-zig, etc., en déploiement).

15.4 Délétion dans l'arbre RN



Pour la délétion, on utilise une technique similaire : procéder comme avec l'arbre binaire de recherche, puis retrogradations en ascendant vers la racine + $O(1)$ rotations (trois au plus) à la fin.
On enlève un nœud y : remplacement par null (si aucun enfant) ou par l'enfant non-null. Ce dernier peut être de rang trop petit (nœud noir remplacé par nœud noir x).

Plusieurs cas :

- ⋆ Cas 0 : nœud rouge x : rétrogradation — il devient noir, et rien plus à faire
- ⋆ Cas 1 : nœud noir avec une sœur noire ; il faut aussi vérifier la couleur des enfants de la sœur (les neveux)
- ⋆ Cas 2 : nœud noir avec une sœur rouge

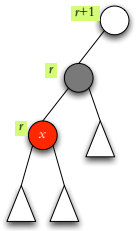
15.5 Arbre RN : temps de calcul des opérations

Recherche	**Insertion**	**Suppression**
$O(h)$	$O(h)$ recherche+ $O(h)$ promotions+ $O(1)$ rotations	$O(h)$ successeur + $O(h)$ rétrogradations + $O(1)$ rotations

Par Théorème 15.7, la hauteur d'un arbre RN est toujours $h = \Theta(\log n)$, donc toutes les opérations s'exécutent en $O(\log n)$.

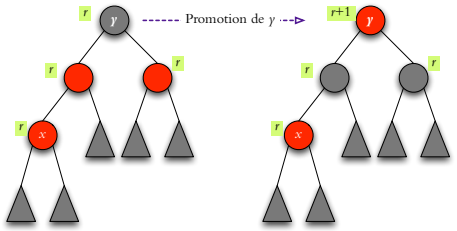
Usage de mémoire : il suffit de stocker la couleur (1 bit) de chaque nœud interne (astuce pour épargner un bit par nœud : échanger les pointeurs gauche ↔ droit pour les nœuds noirs — couleur testée par la comparaison des clés aux enfants)

Cas 0a : parent noir



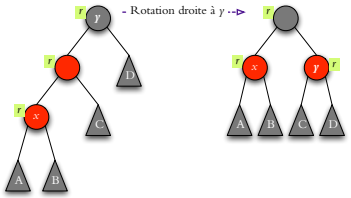
rien à faire

Cas 0b : parent rouge, oncle rouge



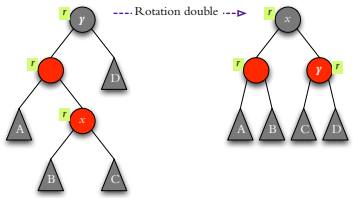
promotion du grandparent, continuer avec $x \leftarrow y$

Cas 1 : parent rouge, oncle noir, zig-zig



une rotation simple

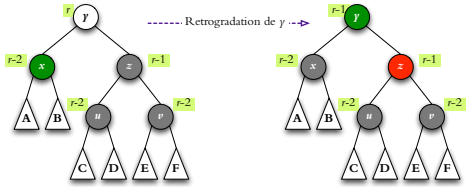
Cas 2 : parent rouge, oncle noir, zig-zag



une rotation double

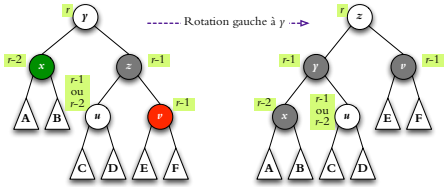
Cas 0 : nœud rouge x — il devient noir, et on arrête

Cas 1a : sœur noire, neveux noirs



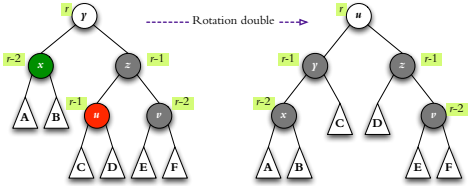
retrogradation du parent, continuer avec $x \leftarrow y$ en cas 0, 1, ou 2

Cas 1b : sœur noire, neveu distant rouge



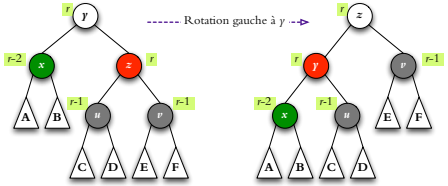
une rotation simple

Cas 1c : sœur noire, neveu proche rouge



une rotation double

Cas 2 : sœur rouge



une rotation simple, continuer avec x en cas 1 (pas de récursion en 1a)